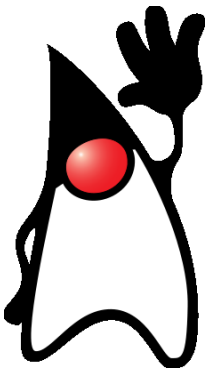
CLASE 7

INTRODUCCIÓN AL LENGUAJE JAVA



TEMAS A VER

- Estructuras de control en JAVA
- Definiciones de clases
- Definiciones de objetos
- Ejemplos

ESTRUCTURAS DE CONTROL

- Encerrar entre {} en caso de incluir varias sentencias.
- Cuando sólo incluye una sentencia, finalizarla con ;

Selección

```
if (condición)
    acción(es) a realizar cuando
    condición es true
else
    acción(es) a realizar cuando
    condición es false
```

Iteración pre-condicional

```
while (condición)
    acción(es) a realizar
cuando
    condición es true
```

Iteración post-condicional

```
do{
    acción(es)
} while (condición)
```

Diferencia do-while y while

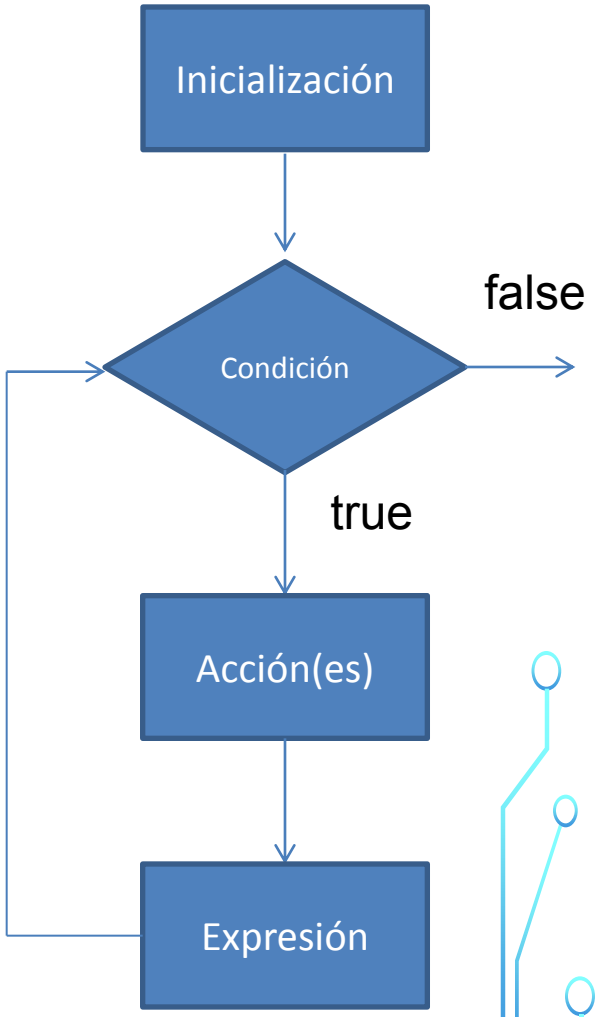
- Ejecuta acción(es) y luego evalúa condición
- Cuando condición es true => ejecuta otra vez acción(es)
- Cuando condición es false => finaliza do

ESTRUCTURAS DE CONTROL

- Encerrar entre {} en caso de incluir varias sentencias.
- Cuando sólo incluye una sentencia, finalizarla con ;

Repetición

for (inicialización; condición; expresión)
    acción(es)



- *Inicialización*: expresión que se ejecuta una vez al comienzo y da valor inicial a la variable índice.
- *Condición*: expresión lógica, se evalúa antes de comenzar una nueva iteración del for; cuando da false termina el for.
- *Expresión*: expresión que se ejecuta al finalizar cada iteración del for (incr. o decr. del índice).

ESTRUCTURAS DE CONTROL. EJEMPLO.

```
int i;
for (i=1; i<= 10;  i++)
   System.out.println(i);
```

¿Qué imprime?

¿Modificar para imprimir pares?

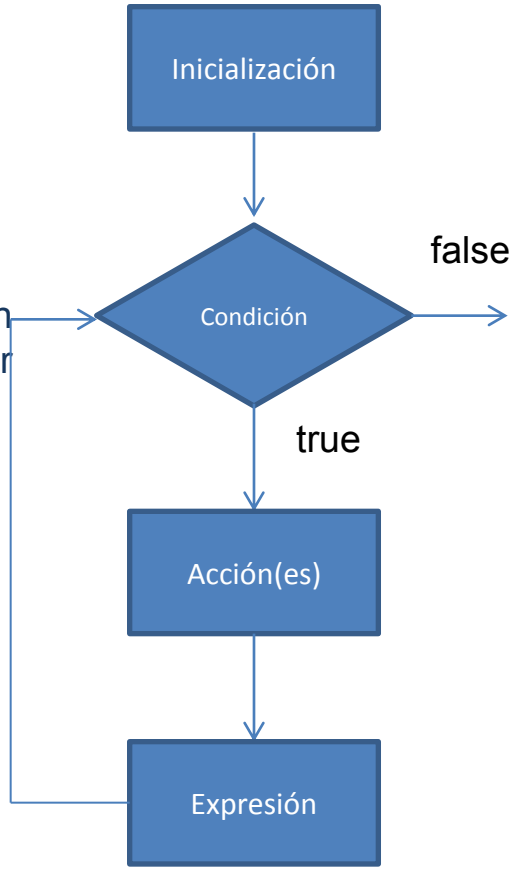
```
int i;
for (i=10; i > 0;  i=i-1)
   System.out.println(i);
```

¿Qué imprime?
¿Es lo mismo poner i-- como expresión?

```
char c;
for (c='a'; c <='z'; c++)
   System.out.println(c);
```

Las variables *char* almacenan el código unicode del carácter
Permite operaciones aritméticas

¿Qué imprime?



ESTRUCTURAS DE CONTROL. EJERCITACIÓN.

- Lea números enteros desde teclado hasta ingresar 999 (no debe procesarse). Informar la suma de los nros impares.

Modificación:

- Lea números enteros desde teclado hasta ingresar 999 (debe procesarse). Informar la suma de los nros impares.

Ayuda: ver tabla de operadores relacionales
Operador igualdad ==
Operador distinto !=

ESTRUCTURAS DE CONTROL - SWITCH

- Permite escoger una entre varias opciones de secuencias de código a ser ejecutado. Su funcionamiento se basa en considerar una expresión que se va secuencialmente comparando con cada valor constante propuesto (o case) como resultado de su evaluación.
- Si el resultado es verdadero (true) se ejecuta el código correspondiente a esa opción.
- Si se desea que el código correspondiente a un sólo un valor propuesto sea ejecutado, es necesario añadir un enunciado de control de flujo **break** para que la estructura switch no evalúe los demás valores propuestos. Al final se añade una opción que recoge cualquier otra posibilidad de resultado de la evaluación de la expresión, bajo la etiqueta **default.**

```
switch (Expresión) {
        case constante char/byte/short/int : secuencia de enunciados
         break; ...
        ..............
         default : secuencia de enunciados }
```

EJEMPLO CON SWITCH

```
class Switch1
   { public static void main (String[] args)
        {Scanner in= new Scanner(System.in);
         System.out.print("Escriba un numero entre 1 y 7: ");
         int day = in.nextInt();
         switch (day)
          {case 1: System.out.println("Domingo"); break;
           case 2: System.out.println("Lunes"); break;
           case 3: System.out.println("Martes"); break;
           case 4: System.out.println("Miercoles"); break;
           case 5: System.out.println("Jueves"); break;
           case 6: System.out.println("Viernes"); break;
           case 7: System.out.println("Sabado"); break;
           default:System.out.println(day + " no es dia"); break;
           }
        }
    }
```

EJEMPLO FOR, ¿QUÉ HACE ESTE CÓDIGO?

```
public class EjemploFor2 {
    public static void main (String[] args) {
        int[] x = new int[10];
        for (int i = 0; i < 10; i++) {
            x[i] = i;
        }
        for (int i = 0; i < 10; i++) {
            System.out.println (x[i]);
        }
    }
}
```

EJEMPLO FOR

```
public class EjemploFor  {
  public static void main (String[] args) {
    if (args.length == 0 ){
      System.out.println ("No hay cadena que transformar");
    }
    else {
      for (int i = 0; i < args.length; i++ ) {
        String s = args[i];
        char[] result = new char[s.length()];
        for (int j = 0; j < s.length(); j++) {
          result[j] = Character.toLowerCase(s.charAt(j));
        }
         for (int j = 0; j < s.length(); j++) {
            System.out.println (result[j]);
            }
      }
    }
}
}
```

El ejemplo muestra un programa que toma una cadena dada como argumento del programa, e imprime una nueva versión de la cadena convirtiendo los caracteres de mayúsculas a minúsculas:

MÉTODOS - RETURN

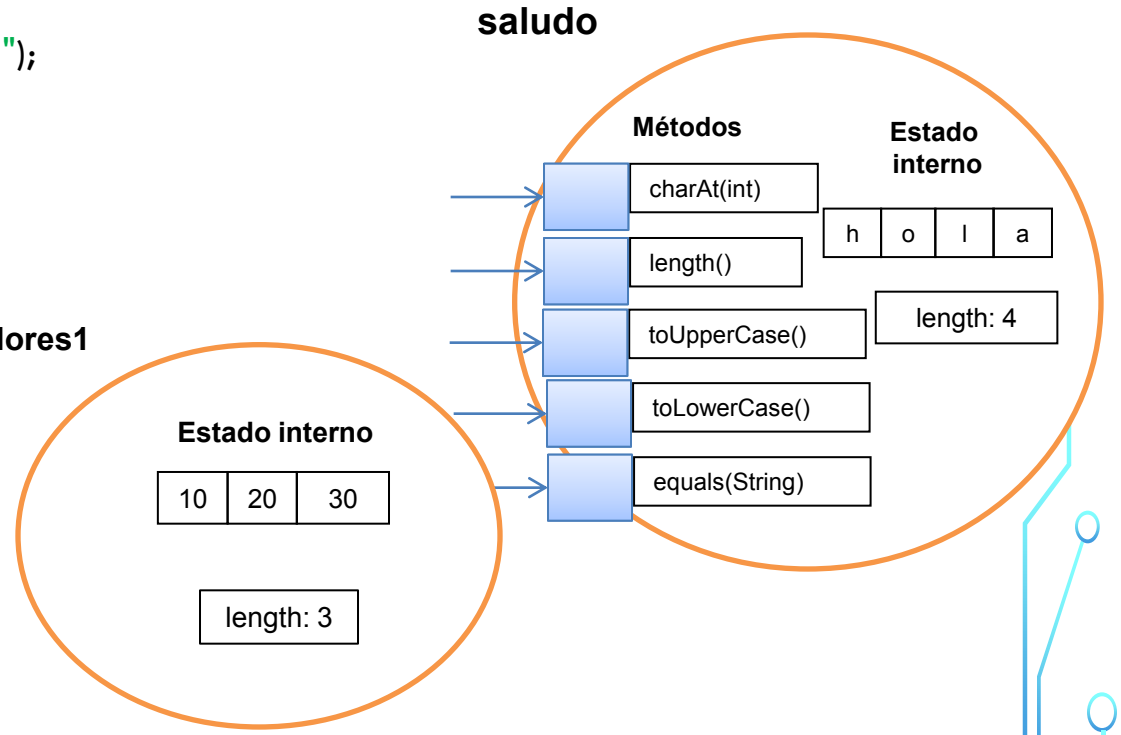
- La palabra **return** puede aparecer en cualquier parte del cuerpo de un método y en múltiples lugares.
- Un método no-vacío debe contener al menos un enunciado return.
- El siguiente ejemplo incluye un método min que retorna el mínimo de dos argumentos:

```
public class EjemploReturn {
   private static int min (int a, int b) {
      if (a < b) return a;
      else return b; }
    public static void main( String[] args) {
        System.out.println("El menor de 5 y 8 es: " + min(5,8));
    }
}
```

Creación de CLASES y OBJETOS en JAVA

OBJETOS EN JAVA

- Java incluye bibliotecas de clases que permiten crear objetos de uso común.
- Ej. clase Scanner, clase Strings, clase Point2D, *clases array (especiales)*, colecciones, …
- En general se crean enviando un mensaje de creación a la clase (new).
- ¿Qué es un string?
  - String saludo = "hola";
  - Otra forma:
    - String saludo = *new* String("hola");
- ¿Qué es un array?
  - int[] valores1 = {10,20,30};
  - Otra forma
    - int [] valores1 = *new int[3]*;
    - valores1[0] = 10;
    - valores1[1] = 20;
    - valores2[2]= 30;



OBJETOS EN JAVA. INSTANCIACIÓN.

- Declarar variable para mantener la referencia:

  NombreDeClase miVariable;

- Enviar a la clase el mensaje de creación y guardar referencia:

  miVariable= new NombreDeClase(valores para inicialización);

- Se puede unir los dos pasos anteriores:

  NombreDeClase miVariable= new NombreDeClase(…);

Ejemplo

String saludo;

saludo= new String("hola");

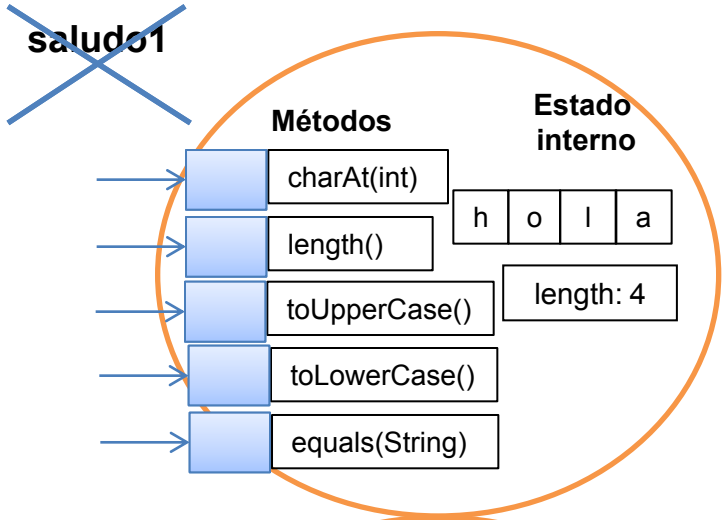
String saludo = new String ("hola");

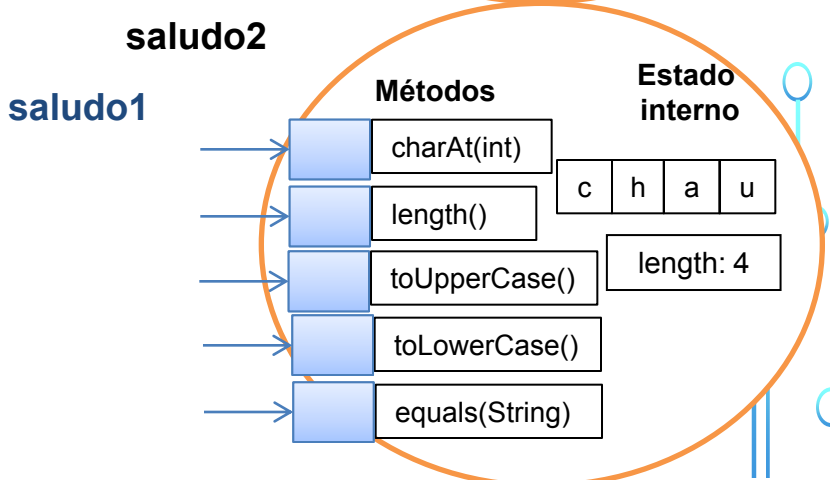
- Secuencia de pasos en la creación:
  - *Alocación de Memoria*. Las variables de instancia se inicializan a valores por defecto.
  - *Inicialización Explícita* (si hubiese) de las variables de instancia.
  - *Ejecución del Constructor* (código para inicializar variables de instancia con los valores que enviamos en el mensaje de creación).
  - *Asignación de la referencia a la variable.*

OBJETOS EN JAVA. REFERENCIAS.

- Referencia a un objeto: ubicación en memoria del objeto.
- Ejemplo
  - String saludo1 = "hola";
- Asignación: copia referencias.
  - String saludo2 = "chau";
  - saludo1 = saludo2;
- Recolector de basura:
  - libera memoria de objetos

  no referenciados.
- Comparación de objetos con == y !=
  - en estos casos comparan referencias
  - String saludo1=new String("hola");
  - String saludo2=new String("hola");
  - System.out.println(saludo1 == saludo2); //false





ENVÍO DE MENSAJE AL OBJETO

Sintaxis

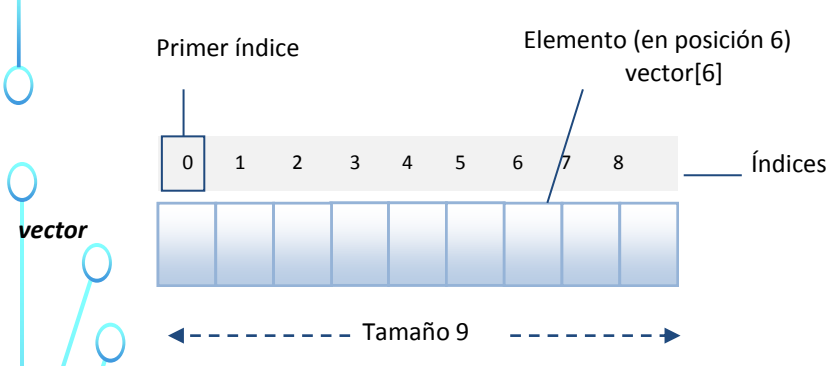
objeto.nombreMétodo(…);

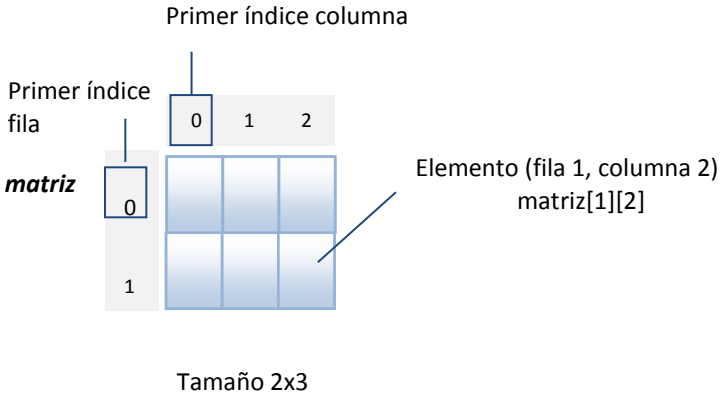
Ejemplo

```
public class DemoEnvioMensaje {
public static void main(String[] args) {
String saludo1 = "hola";
System.out.println(saludo1.length());     //Imprime 4
System.out.println(saludo1.charAt(0));    //Imprime h
    }
}
```

ARREGLOS

- Puede almacenar un número fijo de valores de un mismo tipo primitivo u *objetos del mismo tipo*
- Dimensión física: se establece al crearlo.
- Índice: entero, comenzando desde 0.
- Acceso en forma *directa* a las posiciones.





ARREGLOS

Vector

- **Declaración**

TipoElemento [] nombreVariable*;*

- **Creación**

nombreVariable = new TipoElemento[DIMF];

- **Acceso a elemento**

nombreVariable [posición]

- **Declaración + Inicialización**

TipoElemento [] nombreVariable = {valor0, valor1, … , valorN};

Matriz

- **Declaración**

TipoElemento [][] nombreVariable;

- **Creación**

nombreVariable = new *TipoElemento [DIMF][DIMC];*

- **Acceso a elemento**

nombreVariable [posFil] [posCol]

- **Declaración + Inicialización**

```
TipoElemento [][] nombreVariable = {
        {valores fila 0}, {valores fila 1} , … ,
        {valores fila N}
};
```

ARREGLOS. VECTORES. EJERCITACIÓN.

- Leer desde teclado 10 números enteros. Informar los números ingresados que están por encima del promedio. *(Suponga que los números ingresados son positivos)*.

9 10 9 8 10 2 2 9 8 2 Promedio: 6.9

¿Cuál es el rango para el índice?

ESTRUCTURA DE LA SOLUCIÓN

```
public class DemoVector1 {
        public static void main(String[] args) {
       //Paso 1: Declarar la variable vector de enteros
       //Paso 2: Declarar índice y promedio (iniciarlo)
        //Paso 3: Declarar y crear el scanner
        //Paso 4: Crear el vector para 10 enteros
        //Paso 5: Ingresar 10 números, cargarlos en el vector, ir calculando la suma
         //Paso 6: Calcular el promedio
         //Paso 7: Recorrer el vector, imprimir los números que son mayores que el promedio

    }

}
```

```
import java.util.Scanner;
public class Main
{
    public static void main ()
    {
        Scanner in= new Scanner(System.in);
        int[] vector= new int[10];
        double promedio= 0;
        for (int i= 0;i<10;i++)
        {
            vector[i]= in.nextInt();
            promedio= promedio+ vector[i];
        }
        promedio= promedio/10;
        for (int i= 0;i<10;i++)
        {
            if (vector[i]>promedio)
            {
                System.out.println (vector[i]);
            }
        }
    }
}
```

OBJETOS EN JAVA

- Siga el siguiente programa gráficamente e indique qué imprime. Luego confirme sus resultados ejecutándolo.
- ¿Qué se puede concluir acerca de la *asignación* utilizada con objetos?

```
public class DemoQueImprime {
    public static void main(String[] args) {
        int i;
        int[] valores1 = {10,20,30};
        int[] valores2 = {40,50,60,70};
        valores1 = valores2;
        valores1[0] = 100;
        for (i=0;i<4;i++){
            System.out.println(valores2[i]);
        }
    }
}
```

CLASES EN JAVA

La clase es el núcleo de Java

REPASO:

- Una clase define un nuevo tipo de dato.
- Una clase es una plantilla para un objeto.
- Un objeto es una instancia de una clase.
- A las propiedades y los métodos se les llama miembros de la clase.

CLASES - DECLARACIÓN

Una clase describe la estructura y comportamiento de sus instancias u objetos en términos de variables de instancia y métodos. La accesibilidad de las variables y métodos puede ser explícitamente controlada, permitiendo a la clase actuar como una unidad de encapsulación.

```
class identificador {
  declaraciones del constructor
  declaraciones de métodos
  declaración de métodos estáticos
  declaración de variables de instancia
  declaraciones de variables estáticas
}
```

EJEMPLO

```
class Caja {
   double ancho;
   double alto;
   double fondo;
}
```

Como en las variables o constantes, primero se declara el tipo de dato del objeto, que en este caso es la clase Caja
Observe que la clase empieza con mayúscula

A continuación va el nombre del objeto.
Los nombres de objetos y de variables deben empezar con minúscula

```
Caja objCaja = new Caja();
objCaja.ancho = 20.4;
objCaja.alto = objCaja.fondo = 30.3;
```

La palabra newy a continuación el método Vehiculo() se utilizan para crear el objeto.
El método Caja() se llama constructor por default y crea el objeto con los valores por default

PAUTAS PARA UNA BUENA PROGRAMACIÓN

- Por convención, en Java se utilizan las siguientes reglas para identificadores:
  - Clases e interfaces: Mayúsculas cada palabra
    - class **M**i**C**lase**R**ectangulo {...}
  - Miembros, variables y parámetros: Minúsculas la primera palabra, mayúsculas las siguientes
    - void escribe**E**ntero(int entero) {...}
    - int numLinea;
  - Paquetes: Todo minúsculas (no se suele utilizar más de una palabra)
    - package com.miempresa.proyecto;
  - Constantes: Todo mayúsculas (con subrayados entre palabras)
    - final double **PI** = 3.141592;
    - final int **VELOCIDAD_LUZ** = 299792458;

PÚBLICO, PRIVADO Y PROTEGIDO

public hace que una declaración sea accesible por cualquier clase.

protected hace una declaración accesible por cualquier subclase de la clase que se declara, o a cualquier clase dentro del mismo paquete.

private hace una declaración accesible sólo dentro de la clase en que se declara.

Si no se provee ninguna de estas tres palabras clave, se dice que la declaración tiene accesibilidad por defecto (default accesibility), lo que significa que es accesible por cualquier otra clase dentro del mismo paquete.

MÉTODOS VOID

Los métodos de retorno vacío (o métodos void) se declaran con tipo de retorno void.

El cuerpo de un método void simplemente realiza un procesamiento, que tendrá el efecto colateral de cambiar el estado del objeto para el cual fue invocado, y termina sin explícitamente retornar un valor, aun cuando puede usarse el enunciado return sin argumento.

Ejemplo:

```
void f(){
a = 4; // Supóngase que a es una variable de instancia
return;
}
```

PARAMETROS

- Se pueden pasar objetos a los métodos como parámetros
- Un tipo simple se pasa por valor
  - (byte, short, int, long, char, float, double, boolean)
- Un objeto se pasa por referencia
  - (arrays, Strings, etc.)
- Los métodos pueden devolver objetos

CONSTRUCTORES

Un constructor se declara como cualquier método, contando con una lista de parámetros.
Sin embargo, la declaración de un constructor no tiene tipo de retorno, y su nombre debe coincidir con el nombre de la clase dentro de la cual es declarado.
El constructor tiene la siguiente forma:

Nombre del constructor (listadeparámetros) {
Secuencia de enunciados
}

Un constructor puede ser declarado opcionalmente como public, protected o private.

```
Caja( ) {
  ancho = alto = fondo = 0;
}
```

EJEMPLOS

Métodos constructores

Constructores

```
public class ClaseEjemplo{
    private int atributo;

    public ClaseEjemplo(){
    }

    public ClaseEjemplo(int valorAsignar){
        atributo = valorAsignar;
    }
}
```

Constructor por default

Constructor por parámetros

EEJRCICIO

- Generar una clase para representar libros en venta que se caracterizan por: título, cantidad de páginas, nombre de la editorial, el año de edición, el idioma, el nombre del primer autor, el ISBN, su precio en dólares y la cantidad en stock.
- El libro debe saber:
  - Devolver el valor de cada atributo.
  - Modificar el valor de cada atributo.
  - Devolver un su representación en formato String. Repr. "*Java: A Beginner's Guide por Herbert Schildt - 2014 - ISBN: 978-0071809252*"

DEFINICIÓN DE CLASES.

- Sintaxis

```
public class NombreDeClase {
    /* Declaración del estado del objeto*/
    /* Declaración de constructor(es) */
    /* Declaración de métodos que implementan acciones */
 }
```

DECLARACIÓN DEL ESTADO

Ejemplo

- Estado interno:
  - *Datos de tipos primitivos*
  - *Referencias a otros objetos.*

- Anteponer a la declaración la palabra ***private*** *para lograr encapsulamiento (ocultamiento de la información).*
- En la declaración del dato se puede dar un valor inicial.

```
TipoPrimitivo nombreDato;          double precio;
NombreDeClase  nombreDato;     String titulo;
private double precio;
private double precio  = 10.5;
private String titulo = "Java: A Beginner's Guide";
```

Las v.i.s. privadas pueden ser accedidas sólo dentro de la clase que las declara

DECLARACIÓN DEL ESTADO. EJEMPLO.

Clase Libro

```
public class Libro {
   private String titulo;
   private int paginas;
   private String editorial;
   private int añoEdicion;
   private String idioma;
   private String primerAutor;
   private String ISBN;
   private double precio; /*en dolares*/
   private int cantidadEnStock;
}
```

Los datos correspondientes al estado toman un valor por defecto cuando no se inicializan explícitamente.

¿Qué debo hacer si quiero que mis libros tengan por defecto año de edición 2015 e idioma inglés?

DECLARACIÓN DEL COMPORTAMIENTO

- Sintaxis

```
public TipoRetorno nombreMetodo ( lista de parámetros formales ) {
    /* Declaración de variables locales al método */
    /* Cuerpo del método */
}
```

- public: indica que el método forma parte de la interfaz.
- *TipoRetorno*: tipo de dato primitivo / nombre de clase / void (no retorna dato).
- nombreMetodo: verbo seguido de palabras. Convención de nombres.
- Lista de parámetros: datos de tipos primitivos u objetos.
  - TipoPrimitivo nombreParam // NombreClase nombreParam
  - Separación por coma.
  - Pasaje por valor únicamente.
- Declaración de variables locales. Ámbito. Tiempo de vida.
- Cuerpo. Código puede utilizar estado y modificarlo (v.i.)

DECLARACIÓN DEL COMPORTAMIENTO - PARÁMETROS

- Parámetros: únicamente pasaje por valor
  - *Parámetro dato primitivo:*
    - *Copia del parámetro actual .*
    - *Si se modifica el parámetro formal, no altera el parámetro actual.*
  - *Parámetro objeto:*
    - *Copia de la referencia al objeto pasado como parámetro actual.*
    - *Si se modifica el estado interno del objeto, el cambio es visible fuera.*
    - *Si se modifica la referencia, el parámetro actual sigue referenciando al mismo objeto.*
    - *Analogía con punteros vistos.*

DEFINICIÓN DE CLASES. EJEMPLO

Generar una clase para representar libros en venta que se caracterizan por: título, cantidad de paginas, nombre de la editorial, el año de edición, el idioma, el nombre del primer autor, el ISBN, su precio en dólares y la cantidad en stock.
El libro debe saber:

- Devolver el valor de cada atributo.
- Modificar el valor de cada atributo.
- Devolver un su representación en formato String.

Repr. *"Java: A Beginner's Guide por Herbert Schildt - 2014 - ISBN: 978-0071809252"*

```
public class Libro {
   private String titulo;
   private int paginas;
   private String editorial;
   private int añoEdicion;
   private String idioma;
   private String primerAutor;
   private String ISBN;
   private double precio; /*en dolares*/
   private int cantidadEnStock;

    public String getTitulo(){
        return titulo;
    }

    public void setTitulo(String unTitulo){
        titulo = unTitulo;
    }
```

```
    ...
    public int getCantidadEnStock(){
        return cantidadEnStock;
    }
    public void setCantidadEnStock(int unaCantidad){
         cantidadEnStock=unaCantidad;
    }

    public String toString(){
       return (titulo + " por " + primerAutor + " - " +
                añoEdicion + " -  ISBN: " + ISBN );
    }
}
```

INSTANCIACIÓN (CREACIÓN DE OBJETOS)

- Declarar variable para mantener la referencia:

  NombreDeClase miVariable;

- Enviar a la clase el mensaje de creación:

  miVariable= new NombreDeClase();

- Se puede unir los dos pasos anteriores:

  NombreDeClase miVariable= new NombreDeClase();

Ejemplo

```
Libro libro;

libro = new Libro ();

Libro libro = new Libro ();
```

- Secuencia de pasos en la creación:
  - *Alocación de Memoria*. Las variables de instancia se inicializan a valores por defecto.
  - *Inicialización Explícita*. Se ejecuta el código de inicialización explícito en la declaración de las variables de instancia.
  - *Ejecución del Constructor* (lo veremos la próxima clase).
  - *Asignación de la referencia a la variable.*

ENVÍO DE MENSAJE AL OBJETO

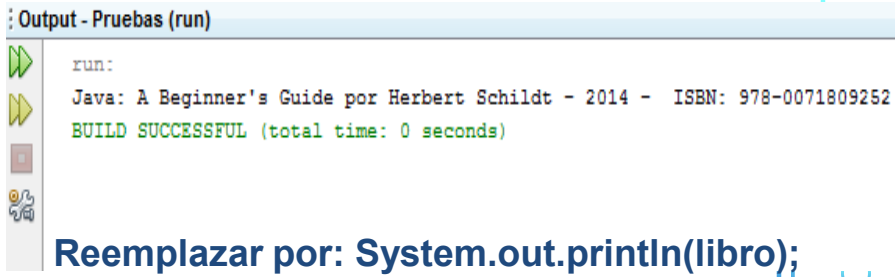
- Sintaxis

  objeto.nombreMétodo(parámetros actuales);

Ejemplo *main*

```
Libro libro = new Libro();
libro.setTitulo("Java: A Beginner's Guide");
libro.setPaginas(699);
libro.setEditorial("Mcgraw-Hill");
libro.setAñoEdicion(2014);
libro.setIdioma("Inglés");
libro.setPrimerAutor("Herbert Schildt");
libro.setISBN("978-0071809252");
libro.setPrecio(21.72);
libro.setCantidadEnStock(100);
System.out.println(libro.toString());
```

DemoLibro.java



Reemplazar por: System.out.println(libro);
Envío automático del mensaje toString al objeto

ACCESO A VARIABLES DE INSTANCIA PRIVATE VS. PUBLIC

- Sintaxis

  objeto.nombreVariableInstancia

- Al declararlas *private* conservamos el encapsulamiento.
  - Sólo los *métodos de instancia* de la clase que declara la v.i. puede accederla.

Ejemplo main anterior System.out.println(libro.ISBN) → No compila

- Algunas clases declaran atributos *public*. Ejemplo arreglos.

```
int[] valores = {10,20,30};
System.out.println(valores.length); //Imprime 3
```

REFERENCIAS Y OBJETOS

- Referencia: ubicación en memoria del objeto.
  - Nos permite interactuar con el objeto.
- Asignación: copia referencias.
- Recolector de basura: libera memoria de objetos no referenciados.

```
public static void main(String[] args) {
    Libro miLibro= new Libro();
    Libro tuLibro = new Libro();
    miLibro.setTitulo("Java: A Beginner's Guide");
    miLibro.setPaginas(699);
    miLibro.setEditorial("Mcgraw-Hill");
    miLibro.setAñoEdicion(2014);
    miLibro.setIdioma("Inglés");
    miLibro.setPrimerAutor("Herbert Schildt");
    miLibro.setISBN("978-0071809252");
    miLibro.setPrecio(21.72);
    miLibro.setCantidadEnStock(100);
    tuLibro = miLibro;
    tuLibro.setPrimerAutor("William Stallings");
    System.out.println(miLibro.getTitulo);
}
```

Constructores

DECLARACIÓN DE CONSTRUCTORES

- Se ejecuta tras alocar el objeto e inicializar las v.i. (por defecto o en la declaración).
- Objetivo: inicialización de v.i.
- Sintaxis

```
public NombreClase( lista de parámetros ) {
    /* Código */
}
```

- Si la clase no declara ningún constructor, Java incluye uno sin parámetros y sin código (*constructor nulo*).
- Instanciación de objeto:

```
NombreClase nombreDato= new NombreClase(lista de parámetros actuales);
```

Ejemplo:
Libro miLibro = new Libro(); //Invoca al *constructor nulo*.

DECLARACIÓN DE CONSTRUCTORES. EJEMPLO

```
public class Libro {
   private String titulo;
   private int paginas;
   private String editorial;
   private int añoEdicion;
   private String idioma;
   private String primerAutor;
   private String ISBN;
   private double precio; /*en dolares*/
   private int cantidadEnStock;
```

```
public Libro(String unTitulo, int unaCantidadPaginas,
             String unaEditorial, int unAñoEdicion,
             String unIdioma,  String unPrimerAutor,
             String unISBN, double unPrecio,
             int unaCantidadStock) {
      titulo = unTitulo;
      paginas = unaCantidadPaginas;
      editorial = unaEditorial;
      añoEdicion= unAñoEdicion;
      idioma= unIdioma;
      primerAutor = unPrimerAutor;
      ISBN =  unISBN;
      precio = unPrecio;
      cantidadEnStock = unaCantidadStock;
   }
…
}
```

DECLARACIÓN DE CONSTRUCTORES. EJEMPLO

- Ejemplo instanciación

```
Libro libro1= new  Libro( "Java: A Beginner's Guide", 699,
                  "Mcgraw-Hill", 2014, "Inglés", "Herbert Schildt",
                  "978-0071809252", 21.72, 100);
```

- ¿Funciona ahora? Libro libro2 = new Libro();

Si el programador generó un constructor,
Java no incluye el constructor nulo.

DECLARACIÓN DE CONSTRUCTORES. SOBRECARGA.

- Puede haber varios constructores para la clase (sobrecarga).
- Java identifica cuál está siendo invocado por el número y tipo de sus parámetros.
- Recomendación: siempre incluir un constructor *sin parámetros*.

¿Qué puedo hacer si quiero que el libro tenga año de edición 2015 e idioma inglés?

Generar otro constructor

DECLARACIÓN DE CONSTRUCTORES. SOBRECARGA. EJEMPLO

```
public class Libro{
…

public Libro(String unTitulo, int
unaCantidadPaginas,
                String unaEditorial, int unAñoEdicion,
                String unIdioma,  String unPrimerAutor,
                String unISBN, double unPrecio,
                int unaCantidadStock) {
        titulo = unTitulo;
        paginas = unaCantidadPaginas;
        editorial = unaEditorial;
        añoEdicion= unAñoEdicion;
        idioma= unIdioma;
        primerAutor = unPrimerAutor;
        ISBN =  unISBN;
        precio = unPrecio;
        cantidadEnStock = unaCantidadStock;
    }
```

```
public Libro(){

  }

public Libro(  String unTitulo, int unaCantidadPaginas,  String unaEditorial,
                String unPrimerAutor, String unISBN, double unPrecio,
                int unaCantidadStock){
        titulo = unTitulo;
        paginas = unaCantidadPaginas;
        editorial = unaEditorial;
        añoEdicion= 2015;
        idioma= "Inglés";
        primerAutor = unPrimerAutor;
        ISBN =  unISBN;
        precio = unPrecio;
        cantidadEnStock = unaCantidadStock;
    }
  …
}
```

3 constructores distintos

DECLARACIÓN DE CONSTRUCTORES. SOBRECARGA. EJEMPLO

```
public static void main(String[] args) {
    Libro libro1= new  Libro( "Java: A Beginner's Guide", 699,
                    "Mcgraw-Hill", 2014, "Inglés",
                    "Herbert Schildt", "978-0071809252", 21.72, 100);
    Libro libro2= new Libro("Learning Java by Building Android Games", 392 ,
                     "CreateSpace Independent Publishing",
                     "John Horton", "978-1512108347", 31.77, 100);
    System.out.println(libro1.getISBN());
    System.out.println(libro2.getEditorial());
    System.out.println(libro2.getIdioma());
    Libro libro3= new Libro();
 }
```

¿Funciona?

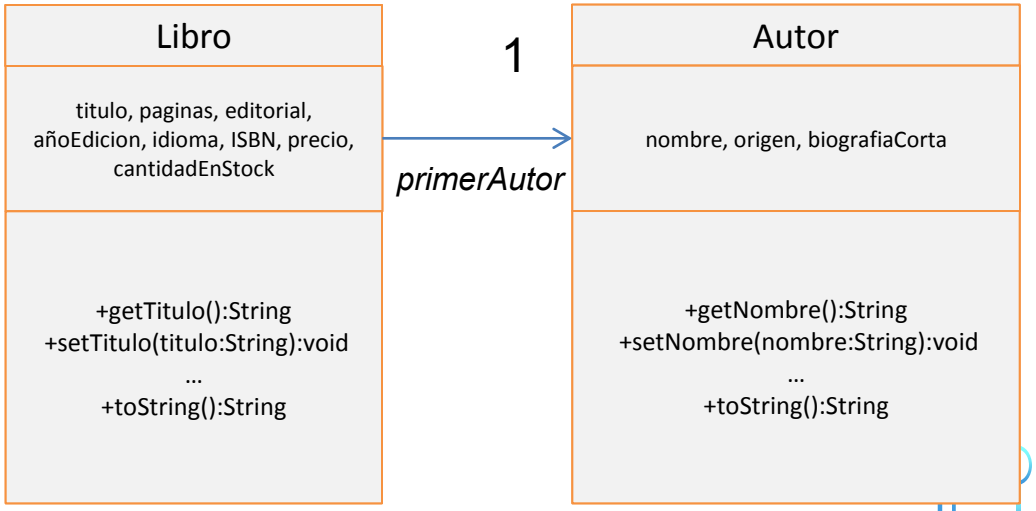
Relaciones entre objetos

RELACIONES ENTRE OBJETOS

- Los objetos necesitan relacionarse con otros para poder interactuar (enviándose mensajes) y así alcanzar objetivos comunes.
- Diseño de SW OO: averiguar cuáles son los objetos del sistema y *las relaciones entre los objetos.*

Ejemplo

Representar libros en venta
que se caracterizan por … primer autor …
El libro debe saber: …
devolver su representación en formato String.
El autor de un libro se caracteriza por
su nombre, origen y una biografía corta.



RELACIONES ENTRE OBJETOS. EJEMPLO

```
public class Autor {
   private String nombre;
   private String origen;
   private String biografiaCorta;

    public Autor (String unNombre, String unOrigen,
                  String unaBiografia){
      nombre=unNombre;
      origen=unOrigen;
      biografiaCorta=unaBiografia;

   }
    public String getNombre(){
        return nombre;
    }
    public String getOrigen(){
        return origen;
    }
    public String getBiografiaCorta(){
        return biografiaCorta;
    }

    public void setNombre(String unNombre){
        nombre=unNombre;
    }

    public void setOrigen(String unOrigen){
        origen=unOrigen;
    }

    public void setBiografiaCorta(String unaBiografia){
        biografiaCorta=unaBiografia;
    }

}
```

```
public class Libro {
   …
   private Autor primerAutor;  /* ref. a un  objeto */

   …
   public Libro(String unTitulo, int unaCantidadPaginas,
                String unaEditorial, int unAñoEdicion,
                String unIdioma,  Autor unPrimerAutor,
                String unISBN, double unPrecio,
                int unaCantidadStock){
        ….
   }

   /* idem 2do constructor*/

   public Autor getPrimerAutor(){
       return primerAutor;
   }
```

```
public void setPrimerAutor(Autor unPrimerAutor){
     primerAutor=unPrimerAutor;
  }

  public String toString(){
       return (titulo + " por " + primerAutor.getNombre() + " -
" + añoEdicion + " -  ISBN: " + ISBN );
   }

}
```

~~primerAutor.nombre~~

Ejemplo

Java: A Beginner's Guide por *Herbert Schildt* – *2014* – ISBN: *978-0071809252*

RELACIONES ENTRE OBJETOS. EJEMPLO

```
public class DemoRelacionesObjetos {

   public static void main(String[] args) {
       Autor autor = new Autor( "Herbert Schildt", "Chicago, Illinois, Estados Unidos",
               "Herbert Schildt is an American computing author, programmer and musician.");
       Libro libro1= new  Libro(  "Java: A Beginner's Guide", 699,  "Mcgraw-Hill",
                2014, "Inglés",  autor, "978-0071809252", 21.72, 100);
       System.out.println(libro1.getAutor());
    }

}
```

Referencia “This”

LA REFERENCIA THIS

- Dentro de un *método de instancia* o de un *constructor,* la referencia *this* representa al objeto que recibió el mensaje o el objeto que está siendo instanciado respectivamente.

- Uso:

  a) Los parámetros del método/constructor que se ejecuta actualmente tienen el mismo nombre que las variables de instancia del objeto. Para referirse a las variables de la instancia se utiliza *this.nombreVariableInstancia*.

```
public class Libro {
    private String titulo;
    private int paginas;
    private String editorial;
    private int añoEdicion;
    private String idioma;
    private Autor primerAutor;
    private String ISBN;
    private double precio;
    private int cantidadEnStock;
```

```
public Libro(  String titulo, int paginas,  String editorial,
   int añoEdicion, String idioma,  Autor primerAutor,
   String ISBN, double precio,  int cantidadEnStock){
      this.titulo= titulo;
      this.paginas= paginas;
      this.editorial= editorial;
      this.añoEdicion= añoEdicion;
      this.idioma= idioma;
      this.primerAutor= primerAutor;
      this.ISBN= ISBN;
      this.precio= precio;
      this.cantidadEnStock= cantidadEnStock;
   }
```

```
public void setTitulo(String titulo){
    this.titulo = titulo;
}
```

LA REFERENCIA THIS

- Uso:

b) El objeto receptor del mensaje o el objeto que está siendo construido debe enviarse mensajes a sí mismo, ej. para desencadenar la ejecución de métodos más simples. Para enviarse un mensaje a sí mismo hacer this.nombreMetodo(parámetros)

```
public class Libro {
   …
   public Libro(  String titulo, int paginas,  String editorial, int añoEdicion, String idioma,
                  Autor primerAutor, String ISBN, double precio,  int cantidadEnStock){
      this.setTitulo(titulo);
      this.setPaginas(paginas);
      …
   }
  public String toString(){
     return (this.getTitulo() + " por " + this.getPrimerAutor().getNombre() + " - " +
            this.getAñoEdicion() + " -  ISBN: " + this.getISBN() );
   }
}
```

LA REFERENCIA THIS

- Uso:
  - c) Invocar desde un constructor a otro, ej. para evitar repetir código. Para invocar a un segundo constructor hacer this(parámetros)

```
public Libro(  String titulo, int paginas,  String editorial,
   int añoEdicion, String idioma,  Autor primerAutor,
   String ISBN, double precio,  int cantidadEnStock){
      this.titulo= titulo;
      this.paginas= paginas;
      this.editorial= editorial;
      this.añoEdicion= añoEdicion;
      this.idioma= idioma;
      this.primerAutor= primerAutor;
      this.ISBN= ISBN;
      this.precio= precio;
      this.cantidadEnStock= cantidadEnStock;
   }
```

```
 public Libro(  String titulo, int paginas,  String editorial, Autor
primerAutor, String ISBN, double precio,  int cantidadEnStock){
        this.titulo = titulo;
        this.paginas = paginas;
        this.editorial = editorial;
        this.añoEdicion= 2015;
        this.idioma= "Inglés";
        this.primerAutor = primerAutor;
        this.ISBN =  ISBN;
        this.precio = precio;
        this.cantidadEnStock = cantidadEnStock;
    }
```

Código repetido: sería mejor invocar al 1er constructor

LA REFERENCIA THIS

- Uso:

  c) Invocar desde un constructor a otro, ej. para evitar repetir código. Para invocar a un segundo constructor hacer this(parámetros)

```
public Libro(  String titulo, int paginas,  String editorial,
   int añoEdicion, String idioma,  Autor primerAutor,
   String ISBN, double precio,  int cantidadEnStock){
      this.titulo= titulo;
      this.paginas= paginas;
      this.editorial= editorial;
      this.añoEdicion= añoEdicion;
      this.idioma= idioma;
      this.primerAutor= primerAutor;
      this.ISBN= ISBN;
      this.precio= precio;
      this.cantidadEnStock= cantidadEnStock;
   }
```

```
public Libro(  String titulo, int paginas,  String editorial, Autor
primerAutor, String ISBN, double precio,  int cantidadEnStock){

      this( titulo, paginas,  editorial, 2015, "inglés",
            primerAutor, ISBN, precio, cantidadEnStock);

   }
```

Restricción: la invocación a otro constructor debe ser la primera línea de código